Digital Image Processing

تمارين تابع للمحاضرة الرابعة والخامسة
Image Enhancement

د. عيسى الغنام   د. إياد حاتم
الفصل الصيفي 2023

Let x:old image

S:new image

S=x+c where c is constant value

>> S=imadd(x,50);// ----------------brighter image

S=x-c where c is constant value

>> S=imsubtract(x,50);// ------------darker image

We can add 2 image :

>> S=x+y;//---------- (where x and y 2 image have same size)

- **Example 1:**

  the following matrix represents the pixels values of an 8-bit image (r) , apply negative transform and find the resulting image pixel values.

  solution:

  L= 28 = 256

  s=L-1-r

  s =255-r

  Apply this transform to each pixel to find the negative

Image (r)

100	110	90	95
98	140	145	135
89	90	88	85
102	105	99	115

Image (s)

155	145	165	160
157	115	110	120
166	165	167	170
153	150	156	140

- **Exercise:**

  the following matrix represents the pixels values of a 5-bit image (r) , apply negative transform and find the resulting image pixel values.

  solution:

Image (r)

21	26	29	30
19	21	20	30
16	16	26	31
19	18	27	23

Image (s)

- The negative of an image can be obtained also with function imcomplement:
  g = imcomplement (f);

Logarithmic transformations are implemented using the expression:

g = c * log (1 + double (f))

But this function changes the data class of the image to double, so another sentence to return it back to uint8 should be done:

gs = im2uint8 (mat2gray(g));

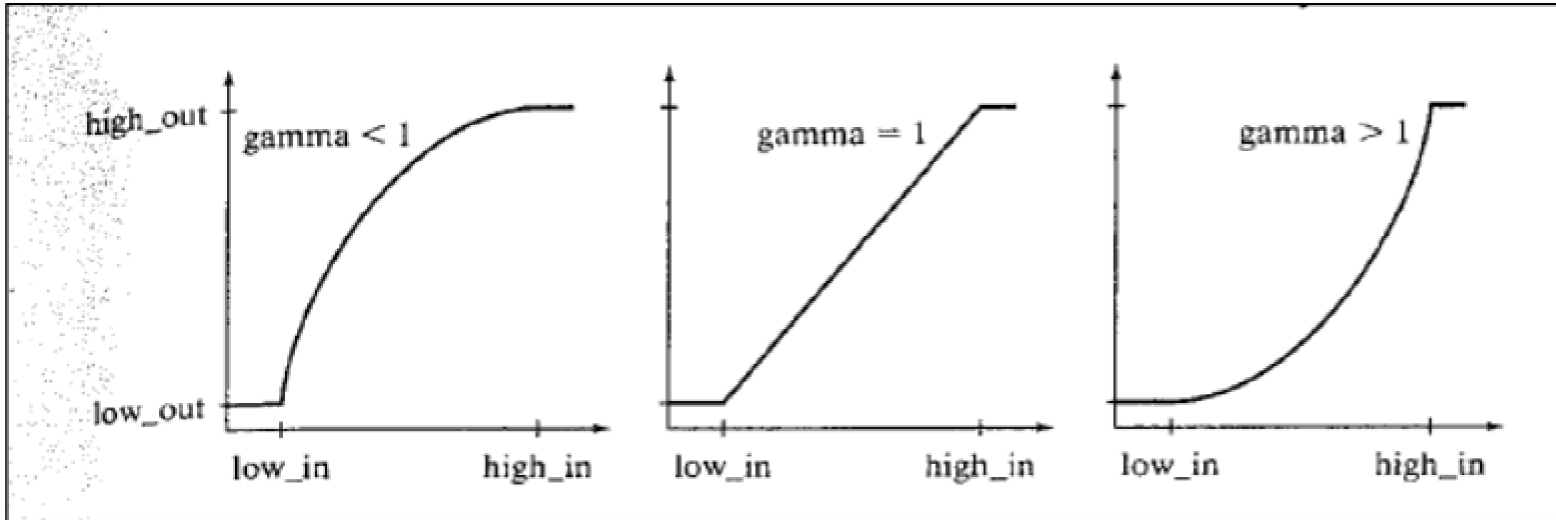
Use of mat2gray brings the values to the range [0 1] and im2uint8 brings them to the range [0 255]

```
>> f=imread('baby.jpg');
>> g = log(1 + double(f));
>> gs = im2uint8(mat2gray(g));
>> imshow(f), figure, imshow (g), figure, imshow(gs);
```

f

g

gs



a b c
FIGURE 3.2 The various mappings available in function imadjust.

Function imadjust is the basic IPT tool for intensity transformations of gray-scale images. It has the syntax:

>> g = imadjust (f, [low_in high_in], [low_out high_out], gamma)

This function maps the intensity values in image f to new values in g, such that values between **low_in** and **high_in** map to values between **low_out** and **high_out.**

Values below **low_in** and above **high_in** are clipped; that is values below **low_in** map to **low_out**, and those above **high_in** map to **high_out**.

- The input image can be of class uint8, uint16, or double, and the output image has the same class as the input.
  All inputs to function **imadjust**, other than **f**, are specified as values between 0 and 1, regardless of the class of **f**.
- If **f** is of class uint8, imadjust multiplies the value supplied by 255 to determine the actual values to use; if **f** is of class uint16, the values are multiplied by 65535.
  Using the empty matrix ([ ]) for [**low_in high_in**] of for [**low_out high_out**] results in the default values [0 1].
- If **high_out** is less than **low_out**, the output intensity is reversed .

- Parameter **gamma** specifies the shape of the curve that maps the intensity values of **f** to create **g**.
- If **gamma** is less than 1, the mapping is weighted toward higher (brighter) output values, as fig 3.2 (a) shows.
- If **gamma** is greater than 1, the mapping is weighted toward lower (darker) output values.
- If it is omitted from the function arguments, **gamma** defaults to 1 (linear mapping).

```
>> f = imread ('baby.jpg');
>> g = imadjust (f, [0 1], [1 0]);
>> imshow(f), figure, imshow (g);
```

This Obtaining the negative image

f

g

```
>> g = imadjust (f, [0.5 0.75], [0 1], .5);
>> imshow(f), figure, imshow (g);
```

f

g

```
>> g = imadjust (f, [0.5 0.75], [0.6 1], 0.5);
>> imshow(f), figure, imshow (g);
```

f

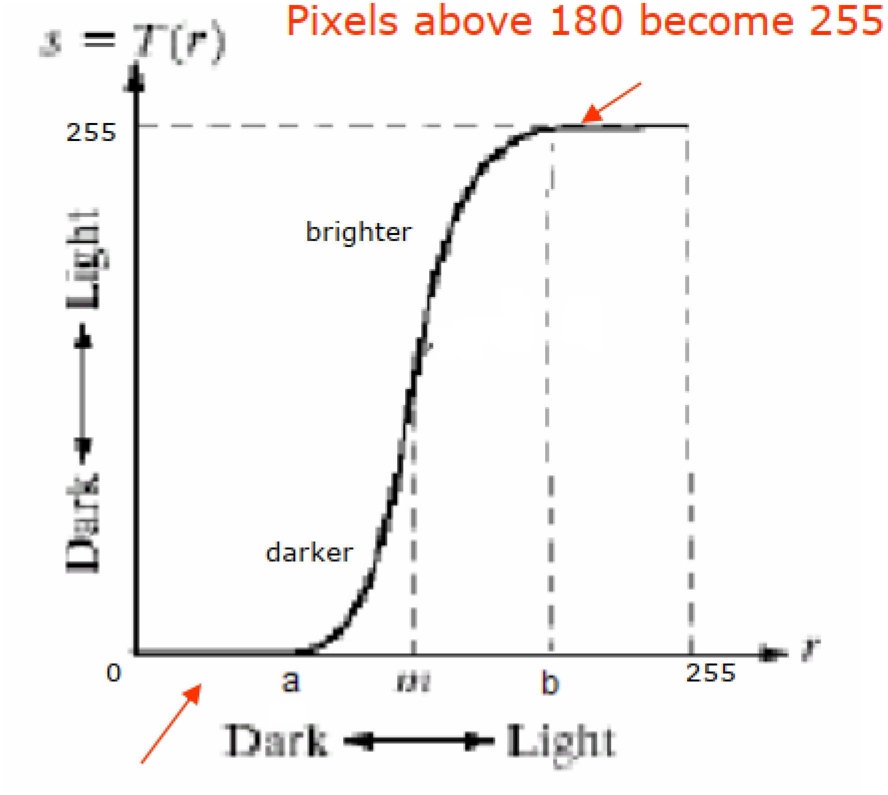
g

```
>> g = imadjust (f, [ ], [ ], 2);
>> imshow(f), figure, imshow (g);
```

f

g

Pixels above 180 become 255



Pixels less than 90 become 0

$$T = \begin{cases} \text{If } r > 180;\ s = 255 \\ \text{If } r < 180 \text{ and } r > 90;\ s = T(r) \\ \text{If } r < 90;\ s = 0 \end{cases}$$

$$s = T(r) = \frac{1}{1 + (m/r)^E}$$

- E controls the slope of the function.
- in the graph, suppose we have the following intensities : a=90, b=180, m=100

This equation is implemented in MATLAB for the entire image as:

```
g = 1./(1 + (m./(double(f) + eps)).^E)
```

Note the use of **eps** to prevent overflow if f has any 0 values.

```
g = 1 ./ (1+ (100 ./(double(f) + eps)) .^ 20);
>> imshow(f), figure, imshow(g);
```

```
g = 1 ./ (1+ (50 ./(double(f) + eps)) .^ 20);
>> imshow(f), figure, imshow(g);
```

```
g = 1 ./ (1+ (150 ./(double(f) + eps)) .^ 20);
>> imshow(f), figure, imshow(g);
```

- **Exercise:**
the following matrix represents the pixels values of a 8-bit image (r) , apply thresholding transform assuming that the threshold m=95, find the resulting image pixel values.

Image (r)

110	120	90	130
91	94	98	200
90	91	99	100
82	96	85	90

Image (s)


```
function a2(x,s)
y=x;
[m n]=size(x);
for i=1:m
for j=1:n
if x(i,j)>= s
y(i,j)=255;
else y(i,j)=0;
end
end
end
figure, imshow(x);
figure, imshow(y);
```

جامعة المنارة
MANARA UNIVERSITY

نهاية المحاضرة